AMENDMENTS TO THE SPECIFICATION

Amend the paragraph found on page 2, lines 5-16 as follows:

An existing architecture is described by ~~describe~~ C.B. Stunkel et al in "Implementing Multidestination Worms in Switch-based Parallel Systems: Architectural Alternatives and their Impact", *Proceedings of the 24th ACM Annual International Symposium on Computer Architecture (ISCA '97)*, June 1997, Denver, CO, USA. They disclose several ways to implement what they call "multidestination worms", which are somewhat similar to multicast frames discussed here. Their preferred solution is a hybrid central buffer/crossbar solution, although the crossbar is really only used to route unicast traffic with low latency. To avoid deadlocks, Stunkel et al take the approach to assure in advance that a frame can be stored in its entirety in the shared buffer. The head-of-frame is not admitted into the switch before this condition is met. This does not imply that, at the time of admittance, there is sufficient space to fit the entire frame, but eventually, through frames exiting the switch, there will be enough space freed to store the entire frame.

Amend the paragraph found on page 8, lines 13-15 as follows:

"Being transmitted" in this sense means that the start-of-frame ~~thestart-of-frame~~ cell of said frame has been transmitted, while the end-of-frame cell has not yet been. A frame being transmitted is also called an *active frame* and the output it is being transmitted on is an *active output*.

Amend the paragraph found on page 9, lines 8-9 as follows:

Note that this equation does not impose a maximum queue size limit for active inputs. Therefore, the output queue size is assumed to be at ~~beat~~ least as large as the shared memory.

Amend the paragraph found on page 10, lines 24-26 as follows:

Consider Fig. 3, where two very long frames, a multicast frame 21 and a unicast frame 22 go through the switch, resulting in outputs 23 and 24. Multicast frame 21 is destined for both outputs 23 and 24 and is assigned a multicast vector indicating the plurality of outputs of the switch, unicast frame 22 is destined for only for output 24.

Amend the paragraph found from page 18, line 25 to page 19, line 3 as follows:

Initially, the output queue is empty and all pointers are *nil*. When a cell arrives, its memory storage address is passed to the destination output queue, along with the input number the cell arrived on, and the cell type (*SoF, CoF, EoF,* wherein *CoF* identifies a continuation cell). Depending on the type of the cell being written, the following three cases must be distinguished:

- *SoF*: If the *OQ* is empty, the cell is entered in the queue at slot *q*, and the head, tail, and read pointers are all set to *q*. If not, the cell is appended at the tail of the queue as indicated by the tail pointer, i.e., the next *SoF* pointer of the tail pointer's entry is set to *q*, and the tail pointer is also set ~~to~~ to *q*. In both cases, the corresponding write pointer is set to *q* also. This write pointer should be *nil*; if it isn't, there is an error condition, because the previous frame from the same input has not been completed

yet. The last *CoF* (or *SoF*, if no *CoF* is present) should be marked as *EoF*, to force an end to the incomplete frame. If the head pointer equals *nil*, it is updated ~~to~~ too.

Amend the paragraph found on page 20, lines 5-9 as follows:

In all three cases it must additionally be checked whether the queue entry just read is being pointed to by the write pointer of the corresponding input (the input the cell being read arrived on). If this is the case, both the write pointer in question and the read pointer must be set to *nil*. The *OQ* also keeps track of the input number of ~~frame~~ frames currently being transmitted.

Amend the paragraph found on page 21, lines 14-17 as follows:

Fig. 8 depicts an example for a shared memory in its upper section and below two variations ~~varitions~~ of an output queue, displaying the output queue size vs. implementation. The central section shows an output queue for $Q < M$, wherein Q is the output queue size and M is the *shared memory size*, and the section at the bottom of the page an output queue of size $Q = M$.